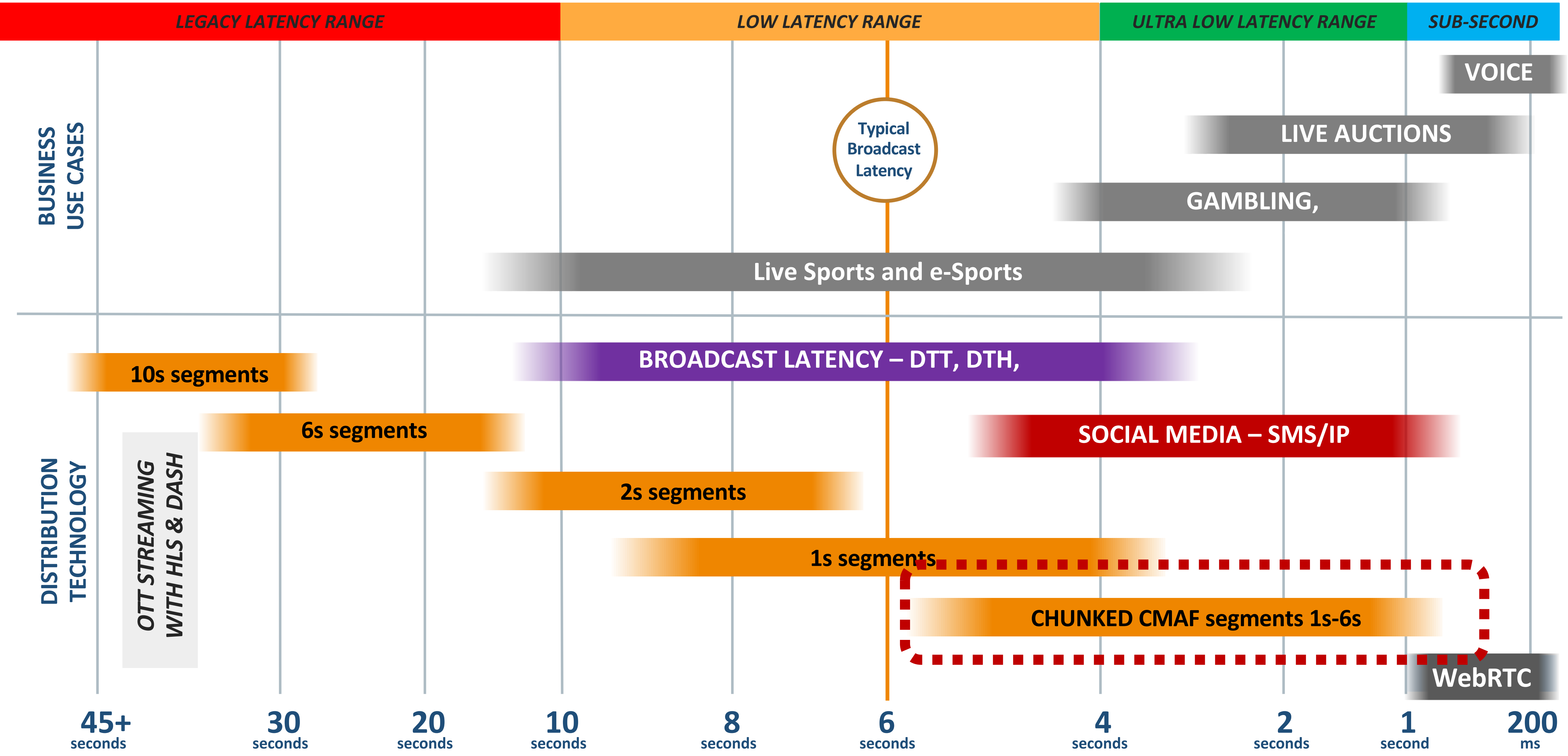
Low Latency Streaming Update

Will Law
Chief Architect
March 2021
For IETF MOPS WG

Akamai
Experience the Edge

Latency Achievable at Scale Via Mainstream CDNs Streaming Technologies



LL-DASH

DASHIF, “Guidelines for Implementation: DASHIF Interoperability Points for ATSC3.0”, July 2016,

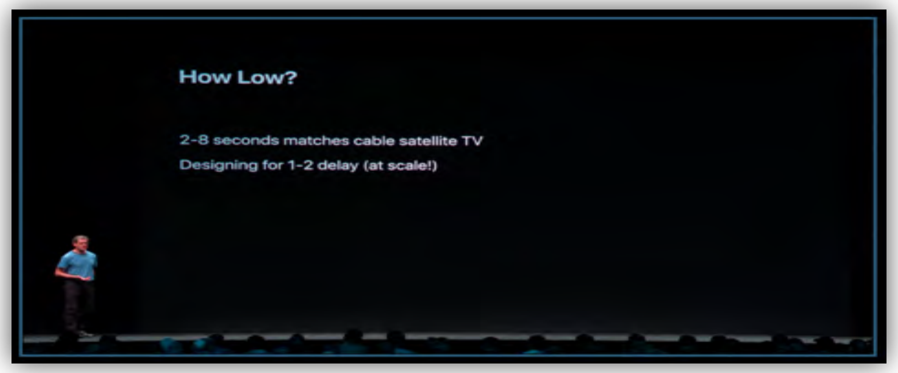
DASHIF, “Guidelines for Low Latency” - https://dashif.org/news/low-latency-dash/ March 2020

DVB-DVB BlueBook A168, ETSI TS 103 285 V1.3.1. Download BlueBook A168

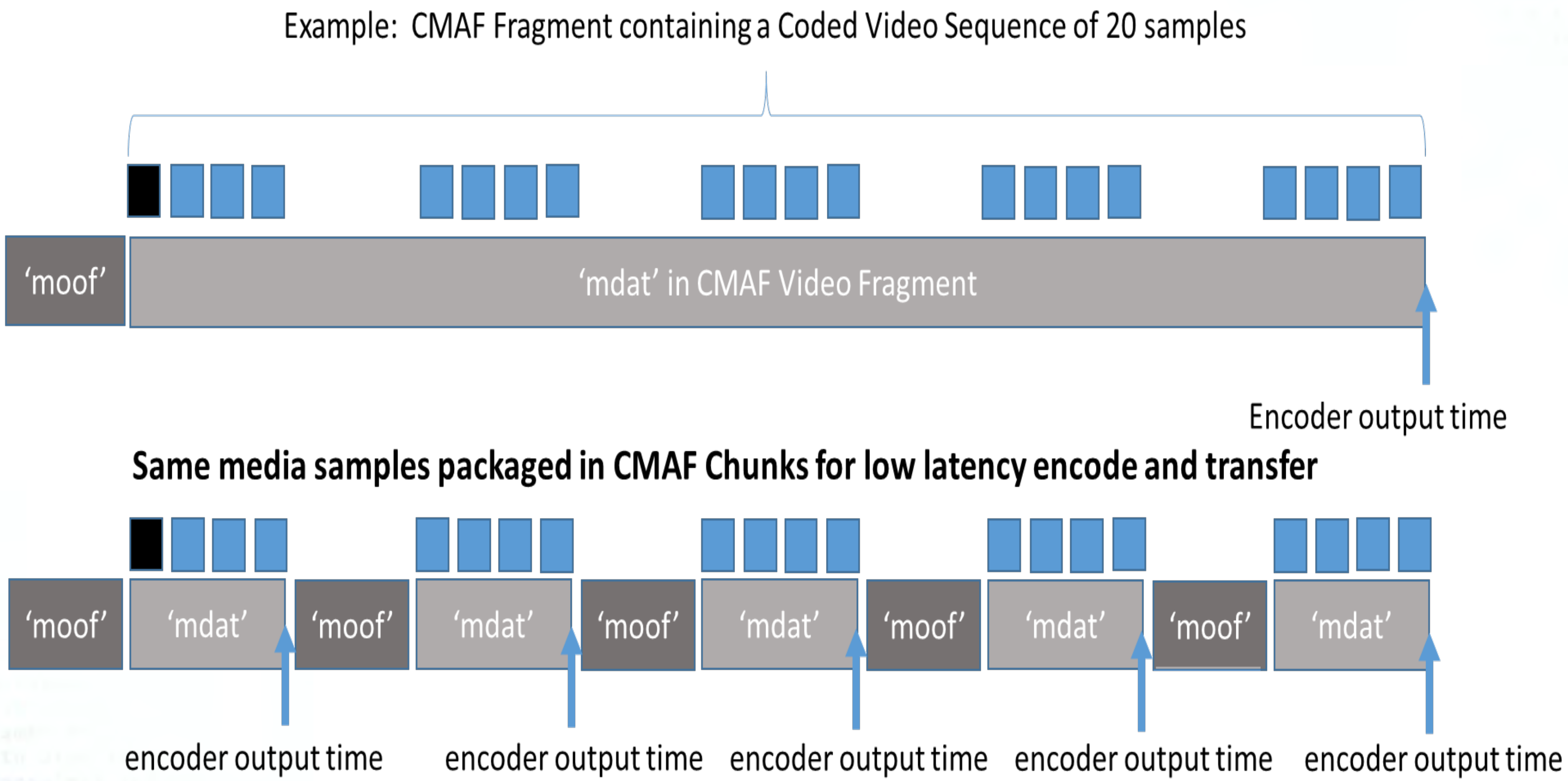
LL-HLS

WWDC – June 5th, 2019 Roger Pantos announced Low-Latency HLS (LL-HLS) https://developer.apple.com/videos/play/wwdc2019/502/

Final spec available at April 30, 2020 https://tools.ietf.org/html/draft-pantos-hls-rfc8216bis-07

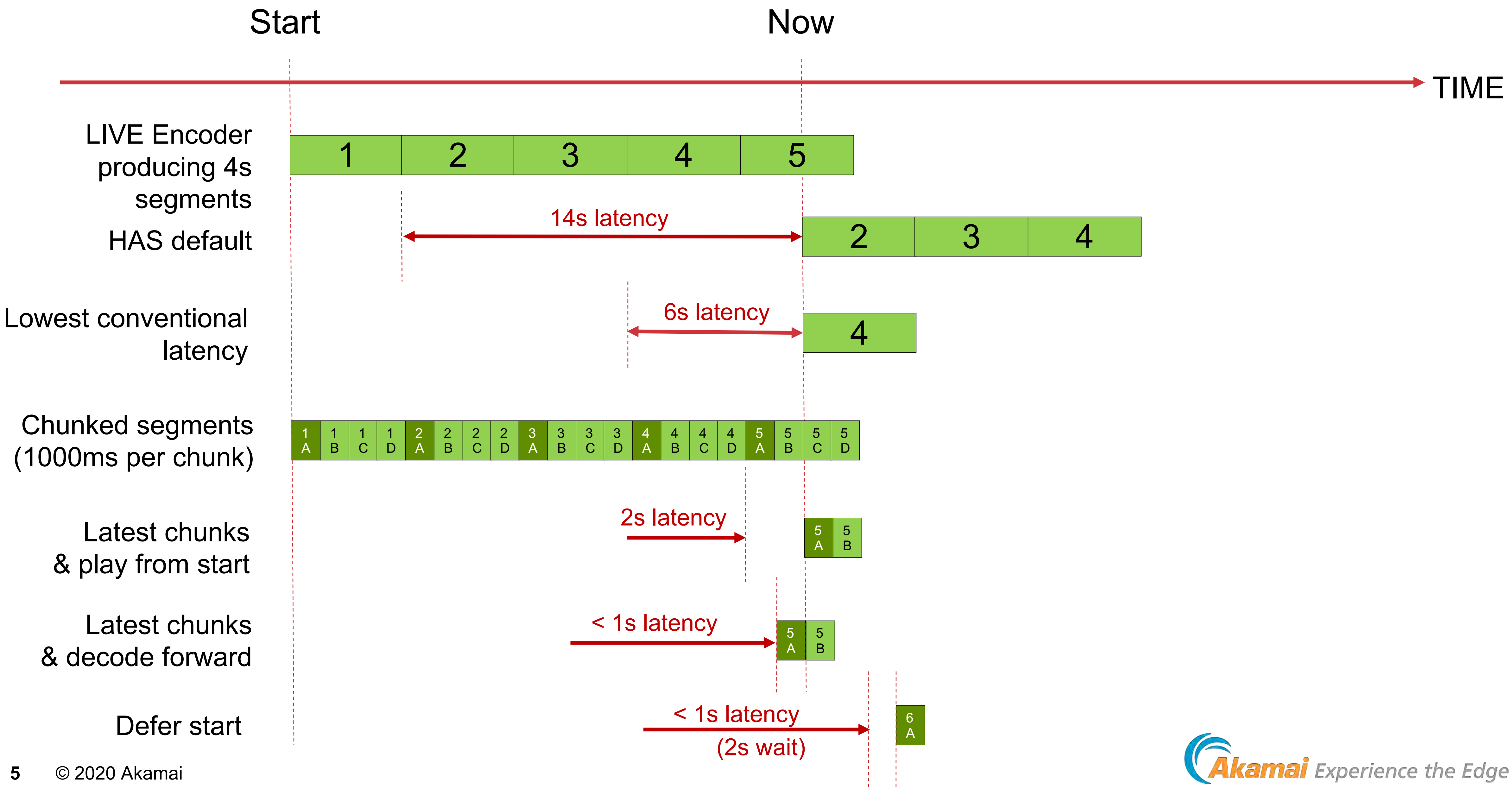


Both solutions use chunked-encoded CMAF

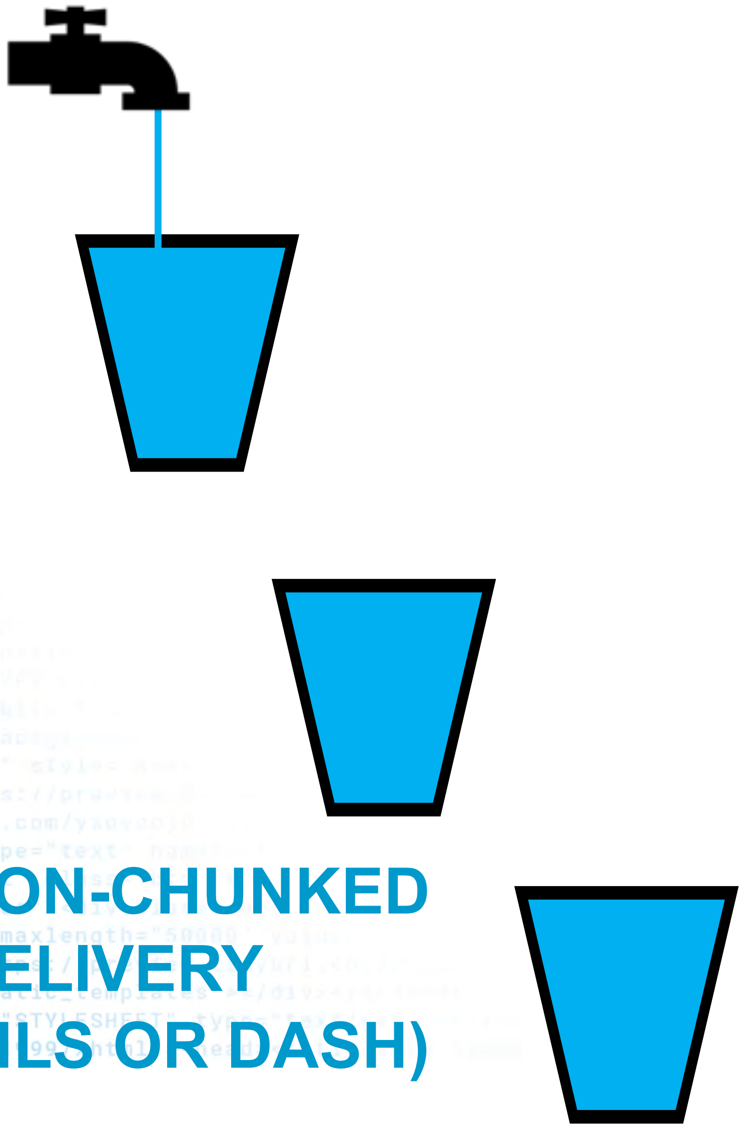


[Image credit deep inside MPEG somewhere, I suspect Kilroy Hughes]

Why does chunking reduce latency?



Different solutions for low latency



NON-CHUNKED DELIVERY (HLS OR DASH)

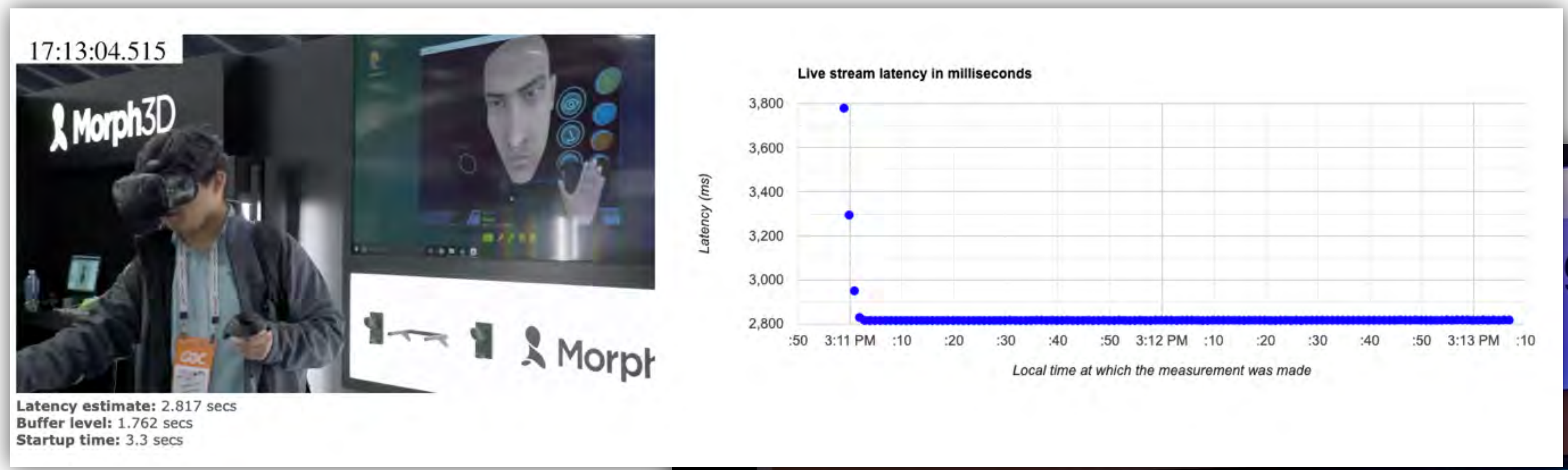
Example low latency DASH manifest

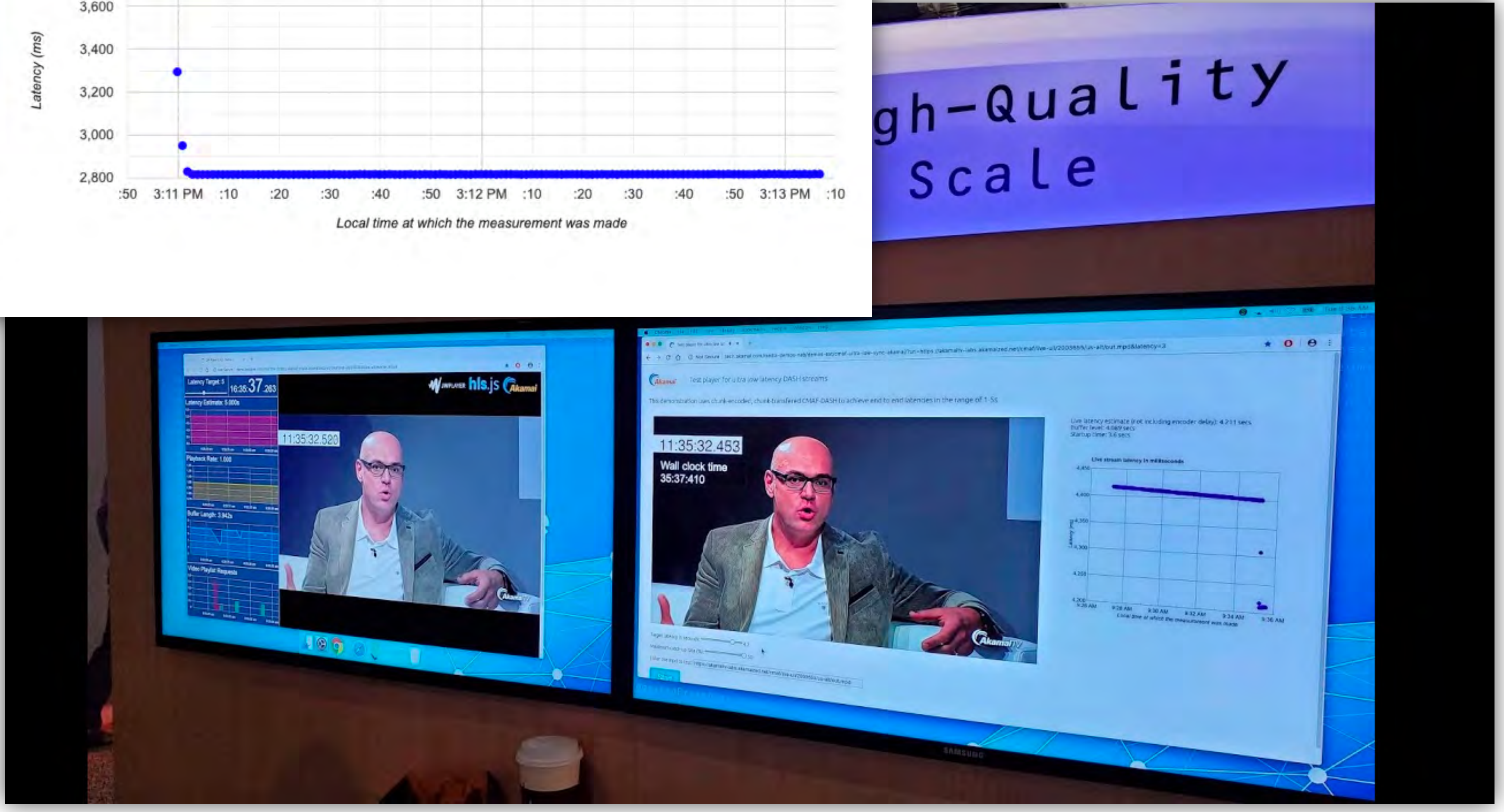

```
<MPD xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance xmlns="urn:mpeg:dash:schema:mpd:2011” xmlns:xlink=http://www.w3.org/1999/xlink
    xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011 http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd”
    profiles="urn:mpeg:dash:profile:isoff-live:2011” type="dynamic” minimumUpdatePeriod="PT20S” availabilityStartTime="2021-02-16T11:12:10.449Z” publishTime="2021-02-
    23T16:29:01.603Z” timeShiftBufferDepth="PT6.0S” maxSegmentDuration="PT2.0S” minBufferTime="PT1.0S"><ServiceDescription id="0"><Latency target="3000"
    referenceId="2"/></ServiceDescription>
    <Period id="0" start="PT0.0S">
            <AdaptationSet id="0" contentType="video" startWithSAP="1" segmentAlignment="true" bitstreamSwitching="true" frameRate="30000/1001" maxWidth="1280"
    maxHeight="720" par="16:9">
                        <Resync dT="33367" type="0"/>
                        <Representation id="0" mimeType="video/mp4" codecs="avc1.64001f" bandwidth="3000000" width="1280" height="720" sar="1:1">
                                <ProducerReferenceTime id="0" inband="true" type="encoder" wallClockTime="2021-02-16T11:12:10.012Z" presentationTime="0">
                                        <UTCTiming schemeIdUri="urn:mpeg:dash:utc:http-xsdate:2014" value="https://time.akamai.com?iso&amp;ms"/>
                                </ProducerReferenceTime>
                                <Resync dT="1001000" type="1"/>
                                <SegmentTemplate timescale="1000000" duration="2002000" availabilityTimeOffset="1.969" availabilityTimeComplete="false"
    initialization="1613473929/init-stream$RepresentationID$.m4s" media="1613473929/chunk-stream_$RepresentationID$-$Number%05d$.m4s" startNumber="1">
                                </SegmentTemplate>
                        </Representation>
            </AdaptationSet>
            …
    </Period>
    <UTCTiming schemeIdUri="urn:mpeg:dash:utc:http-xsdate:2014" value="https://time.akamai.com?iso&amp;ms"/>
</MPD>
```


Example low latency HLS media playlist

```
#EXTM3U
#EXT-X-TARGETDURATION:4
#EXT-X-VERSION:6
#EXT-X-SERVER-CONTROL:CAN-BLOCK-RELOAD=YES,CAN-SKIP-UNTIL=24,PART-HOLD-BACK=3.012
#EXT-X-PART-INF:PART-TARGET=1.004000
#EXT-X-MEDIA-SEQUENCE:236928
#EXT-X-MAP:URI="fileSequence56.mp4"
#EXT-X-PROGRAM-DATE-TIME:2021-02-23T16:34:54.261Z
#EXTINF:4.00000,
fileSequence237476.mp4
#EXTINF:4.00000,
fileSequence237477.mp4
#EXT-X-PART:DURATION=1.00000,INDEPENDENT=YES,URI="lowLatencySeg.mp4?segment=filePart237478.1.mp4"
#EXT-X-PART:DURATION=1.00000,INDEPENDENT=YES,URI="lowLatencySeg.mp4?segment=filePart237478.2.mp4"
#EXT-X-PART:DURATION=1.00000,INDEPENDENT=YES,URI="lowLatencySeg.mp4?segment=filePart237478.3.mp4"
#EXT-X-PART:DURATION=1.00000,INDEPENDENT=YES,URI="lowLatencySeg.mp4?segment=filePart237478.4.mp4"
#EXTINF:4.00000,
fileSequence237478.mp4
#EXT-X-PART:DURATION=1.00000,INDEPENDENT=YES,URI="lowLatencySeg.mp4?segment=filePart237479.1.mp4"
#EXT-X-PART:DURATION=1.00000,INDEPENDENT=YES,URI="lowLatencySeg.mp4?segment=filePart237479.2.mp4"
#EXT-X-PART:DURATION=1.00000,INDEPENDENT=YES,URI="lowLatencySeg.mp4?segment=filePart237479.3.mp4"
#EXT-X-PART:DURATION=1.00000,INDEPENDENT=YES,URI="lowLatencySeg.mp4?segment=filePart237479.4.mp4"
#EXTINF:4.00000,
fileSequence237479.mp4
#EXT-X-PART:DURATION=1.00000,INDEPENDENT=YES,URI="lowLatencySeg.mp4?segment=filePart237480.1.mp4"
#EXT-X-PART:DURATION=1.00000,INDEPENDENT=YES,URI="lowLatencySeg.mp4?segment=filePart237480.2.mp4"
#EXT-X-PRELOAD-HINT:TYPE=PART,URI="lowLatencySeg.mp4?segment=filePart237480.3.mp4"
#EXT-X-RENDITION-REPORT:URI="/cmaf/audio/lowLatencyHLS.m3u8",LAST-MSN=237578,LAST-PART=0
#EXT-X-RENDITION-REPORT:URI="/cmaf/media0/lowLatencyHLS.m3u8",LAST-MSN=236944,LAST-PART=1
#EXT-X-RENDITION-REPORT:URI="/cmaf/media2/lowLatencyHLS.m3u8",LAST-MSN=236944,LAST-PART=0
```

Multiple commercial & open source players now available with 2-3s stable latency






01:51:52.403
SNOW
01:51:52.469
SNOW
01:51:52.469
SNOW
MacBook Air

Commonalities

	LL-HLS	DASH
Require content to be chunk encoded	●	●
Support E2E latencies from 2-10s+	●	●
Backwards compatible with older players	●	●
Cacheable by CDNs	●	●
Support DRM	●	●
Support ad insertion	●	●
Support multiple codec types	●	●
Allow ABR playback	●	●
HTTP delivery	●	●

Differences

	LL-HLS	DASH
Use chunk encoded transfer	🔴	🟢
Describe internal segment structure	🟢	🔴
Require playlist refresh with each chunk	🟢	🔴
Objects always delivered at line speed	🟢	🔴
Support TS segments	🟢	🔴
Require HTTP2 for last mile	🟢	🔴
Require smart origin to modify playlists	🟢	🔴
Deterministic start-up	🟢	🔴

Akamai Experience the Edge

Difference #2: request rates

Let's compare client request rates for a 6s segment with 333ms chunks/parts.

DASH-LL

20 requests/min to CDN edge

LL-HLS

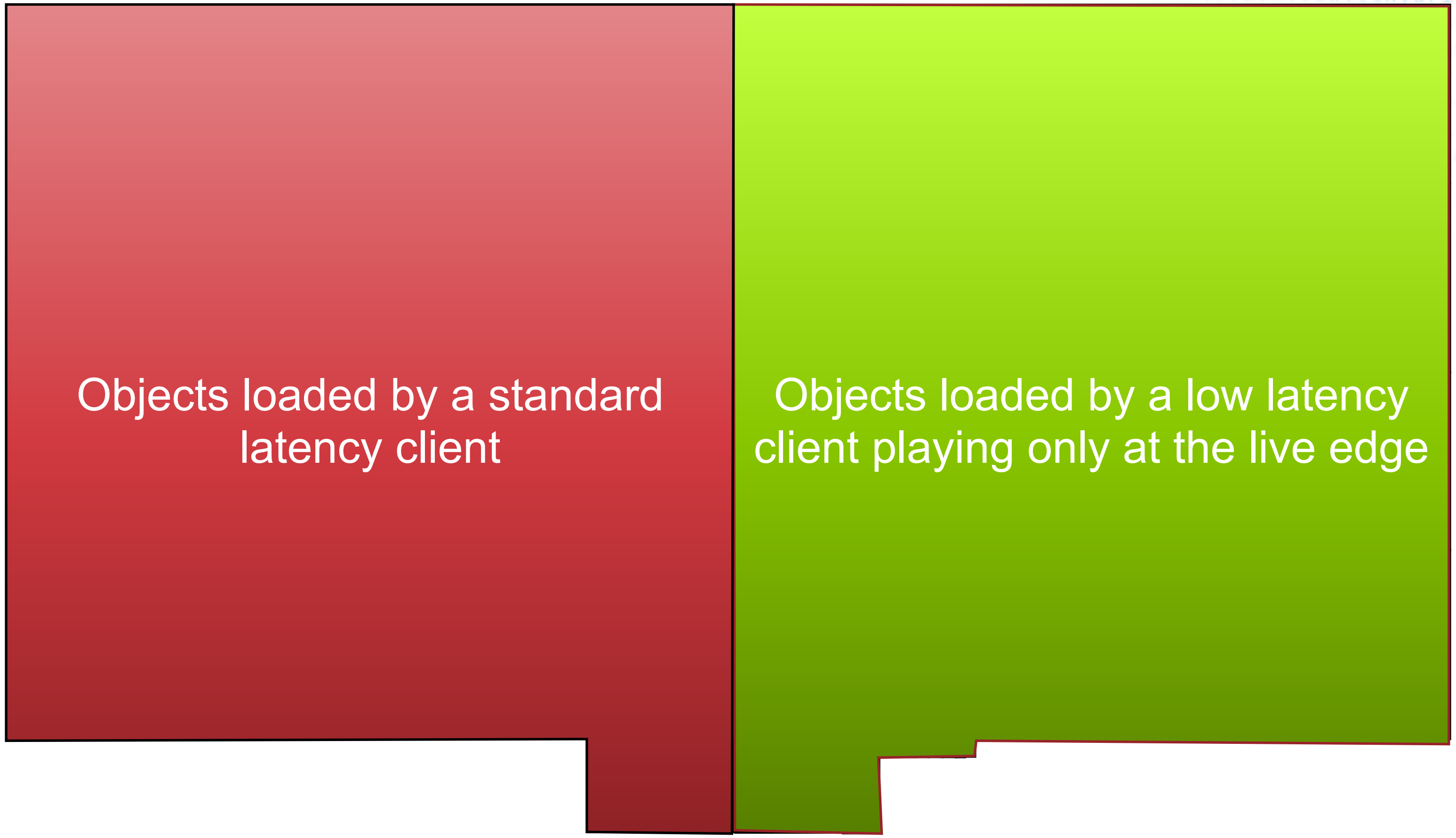
720 requests/min to CDN edge

Using byte-range addressing with LL-HLS to achieve interop with DASH & LL-DASH

LL-HLS – cache footprint

We compare the first 4s of a low latency stream (4Mbps video, 96kbps audio) with discreet part addressing.

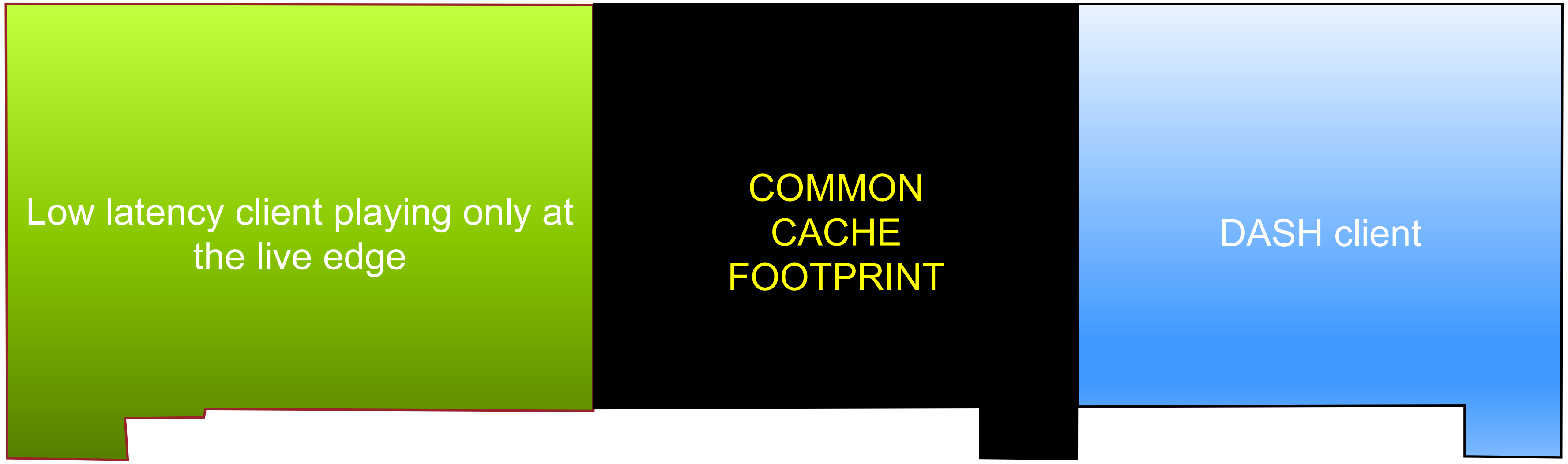
master.m3u8	0.6 KB
video.m3u8	2.5 KB
audio.m3u8	2.5 KB
video.m3u8?_HLS_part=0	2.5 KB
video.m3u8?_HLS_part=1	2.5 KB
video.m3u8?_HLS_part=2	2.5 KB
video.m3u8?_HLS_part=3	2.5 KB
audio.m3u8?_HLS_part=0	2.5 KB
audio.m3u8?_HLS_part=1	2.5 KB
audio.m3u8?_HLS_part=2	2.5 KB
audio.m3u8?_HLS_part=3	2.5 KB
video-init.mp4	0.7 KB
video1.0.mp4	500 KB
video1.1.mp4	500 KB
video1.2.mp4	500 KB
video1.3.mp4	500 KB
video1.mp4	2000 KB
audio-init.mp4	0.6 KB
audio1.0.mp4	12 KB
audio1.1.mp4	12 KB
audio1.2.mp4	12 KB
audio1.3.mp4	12 KB
audio1.mp4	48 KB



LL-HLS + DASH cache footprint

master.m3u8	0.6 KB
video.m3u8	2.5 KB
audio.m3u8	2.5 KB
video.m3u8?_HLS_part=0	2.5 KB
video.m3u8?_HLS_part=1	2.5 KB
video.m3u8?_HLS_part=2	2.5 KB
video.m3u8?_HLS_part=3	2.5 KB
audio.m3u8?_HLS_part=0	2.5 KB
audio.m3u8?_HLS_part=1	2.5 KB
audio.m3u8?_HLS_part=2	2.5 KB
audio.m3u8?_HLS_part=3	2.5 KB
video-init.mp4	0.7 KB
video1.0.mp4	500 KB
video1.1.mp4	500 KB
video1.2.mp4	500 KB
video1.3.mp4	500 KB
video1.mp4	2000 KB
audio-init.mp4	0.6 KB
audio1.0.mp4	12 KB
audio1.1.mp4	12 KB
audio1.2.mp4	12 KB
audio1.3.mp4	12 KB
audio1.mp4	48 KB

manifest.mpd	0.6 KB
video-init.mp4	0.7 KB
video1.mp4	2000 KB
audio-init.mp4	0.6 KB
audio1.mp4	48 KB



Akamai Experience the Edge

LL-HLS Byte range addressing

```
#EXT-X-PART:DURATION=0.500,URI="v1_1-400416425.6.m4s"
#EXT-X-PART:DURATION=0.500,URI="v1_1-400416425.7.m4s"
#EXTINF:4.000,
v1_1-400416425.m4s
#EXT-X-PROGRAM-DATE-TIME:2020-10-02T19:08:27.995Z
#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.0m4s",INDEPENDENT=YES
#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.1m4s"
#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.2m4s"
#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.3m4s"
#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.4m4s"
#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.5.m4s"
#EXT-X-PRELOAD-HINT:TYPE=PART,URI="v1_1-400416426.6.m4s"
```

```
#EXT-X-PART:DURATION=0.500,URI="v1_1-400416425.m4s",BYTERANGE="251022@2079557"
#EXT-X-PART:DURATION=0.500,URI="v1_1-400416425.m4s",BYTERANGE="250203@2330579"
#EXTINF:4.000,
v1_1-400416425.m4s
#EXT-X-PROGRAM-DATE-TIME:2020-10-02T19:08:27.995Z
#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.m4s",BYTERANGE="472180@0",INDEPENDENT=YES
#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.m4s",BYTERANGE="308047@472180"
#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.m4s",BYTERANGE="242094@780227"
#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.m4s",BYTERANGE="223612@1022321"
#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.m4s",BYTERANGE="478404@1245933"
#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.m4s",BYTERANGE="281142@1724337"
#EXT-X-PRELOAD-HINT:TYPE=PART,URI="v1_1-400416426.m4s",BYTERANGE-START=2005479
```

Implications of an open range request

Imagine a segment 4MB in size. If the media playlist describes this hint as shown below, what must the origin do?

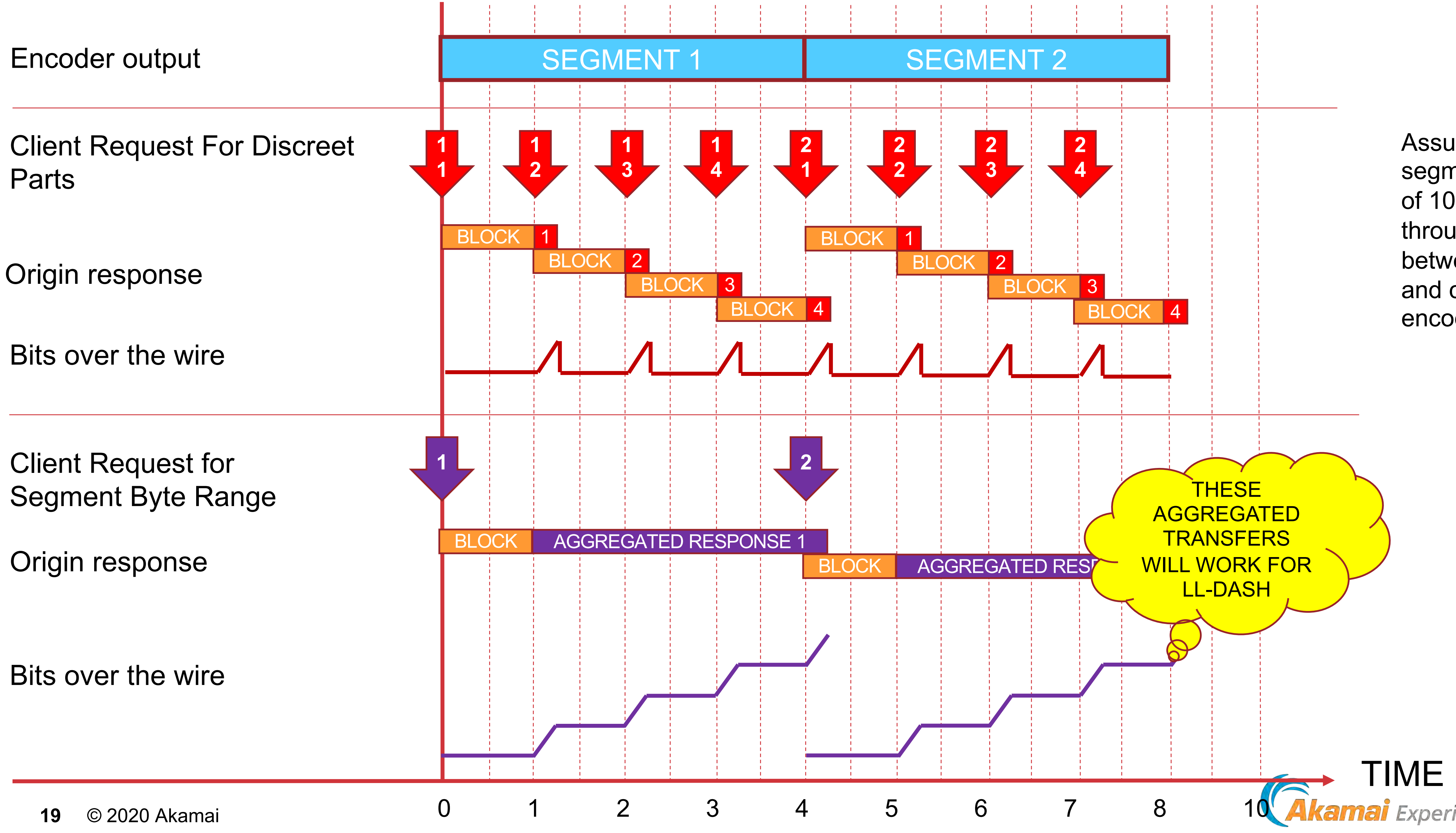
#EXT-X-PRELOAD-HINT:TYPE=PART,URI=”segment1000.m4s",BYTERANGE-START=2005479

According to the HLS spec at https://tools.ietf.org/html/draft-pantos-hls-rfc8216bis-07 , it must “*When processing requests for a URL or a byte range of a URL that includes one or more Partial Segments that are not yet completely available to be sent - such as requests made in response to an EXT-X- PRELOAD-HINT tag - the server MUST refrain from transmitting any bytes belonging to a Partial Segment until all bytes of that Partial Segment can be transmitted at the full speed of the link to the client. If the requested range includes more than one Partial Segment then the server MUST enforce this delivery guarantee for each Partial Segment in turn. This enables the client to perform accurate Adaptive Bit Rate (ABR) measurements.*”

So the origin begins an open-ended response, starting at that offset, and bursting each part as it becomes available.

That single hint request will therefore return the remainder of that segment.

Common workflow – LL-HLS and LL-DASH



Assume 4s segments, parts of 1000ms and throughput between server and client of 4x encoded bitrate.

Start-up request flow

#EXT-X-PART:DURATION=0.500,URI="v1_1-400416425.m4s",BYTERANGE="251022@2079557"

#EXT-X-PART:DURATION=0.500,URI="v1_1-400416425.m4s",BYTERANGE="250203@2330579"

#EXTINF:4.000,

v1_1-400416425.m4s

#EXT-X-PROGRAM-DATE-TIME:2020-10-02T19:08:27.995Z

#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.m4s",BYTERANGE="472180@0",INDEPENDENT=YES

#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.m4s",BYTERANGE="308047@472180"

#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.m4s",BYTERANGE="242094@780227"

#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.m4s",BYTERANGE="223612@1022321"

#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.m4s",BYTERANGE="478404@1245933"

#EXT-X-PART:DURATION=0.500,URI="v1_1-400416426.m4s",BYTERANGE="281142@1724337"

#EXT-X-PRELOAD-HINT:TYPE=PART,URI="v1_1-400416426.m4s",BYTERANGE-START=2005479

Option #1: 7 requests

Option #2: 1 request

GET v1_1-400416426.m4s range=0-472179

GET v1_1-400416426.m4s range=472180-780226

GET v1_1-400416426.m4s range=780227-1022320

GET v1_1-400416426.m4s range=1022321-1245932

GET v1_1-400416426.m4s range=1245932-1724336

GET v1_1-400416426.m4s range=1724336-2005478

GET v1_1-400416426.m4s range=2005479-

This single request will return all the parts in sequence just as if they had been requested individually

Oh dear, now the CDN is confused

Imagine you are an edge server, and you receive a client request for range=0- against an object whose size you do not yet know. Let's imagine its actual size is 1000B and you have the first 100B received at the edge.

Do you

A: wait until you have received an EOF signal and return a 200 response code with content-length 1000 ? or

B: immediately return the 100B you do have in an open-ended 206 response and close the response once the 1000'th byte is delivered?

Both are valid use cases.
How can the client signal the type of response it wishes to receive?

RFC 8673 – to the rescue

- Per this solution, the client should never make an open ended range request if it is expecting an aggregated response from a fixed offset.
- It should instead send a request with a very large number as the last-byte-pos in the range request. 9007199254740991 has been proposed.This equals Number.MAX_SAFE_INTEGER for 64 bit systems)
- This would signal the server (or origin) to begin a 206 response that starts at the requested offset and aggregates over time until the object is completely transferred.

What does steady state look like?

GET / v1_1-7729.m4s HTTP/2
Range: bytes=0-9007199254740991

GET / v1_1-7730.m4s HTTP/2

GET / v1_1-7731.m4s HTTP/2

GET / v1_1-7732.m4s HTTP/2

GET / v1_1-7733.m4s HTTP/2

GET / v1_1-7734.m4s HTTP/2

A LL-HLS client using byte range addressing need only make one request per segment duration for each media type.

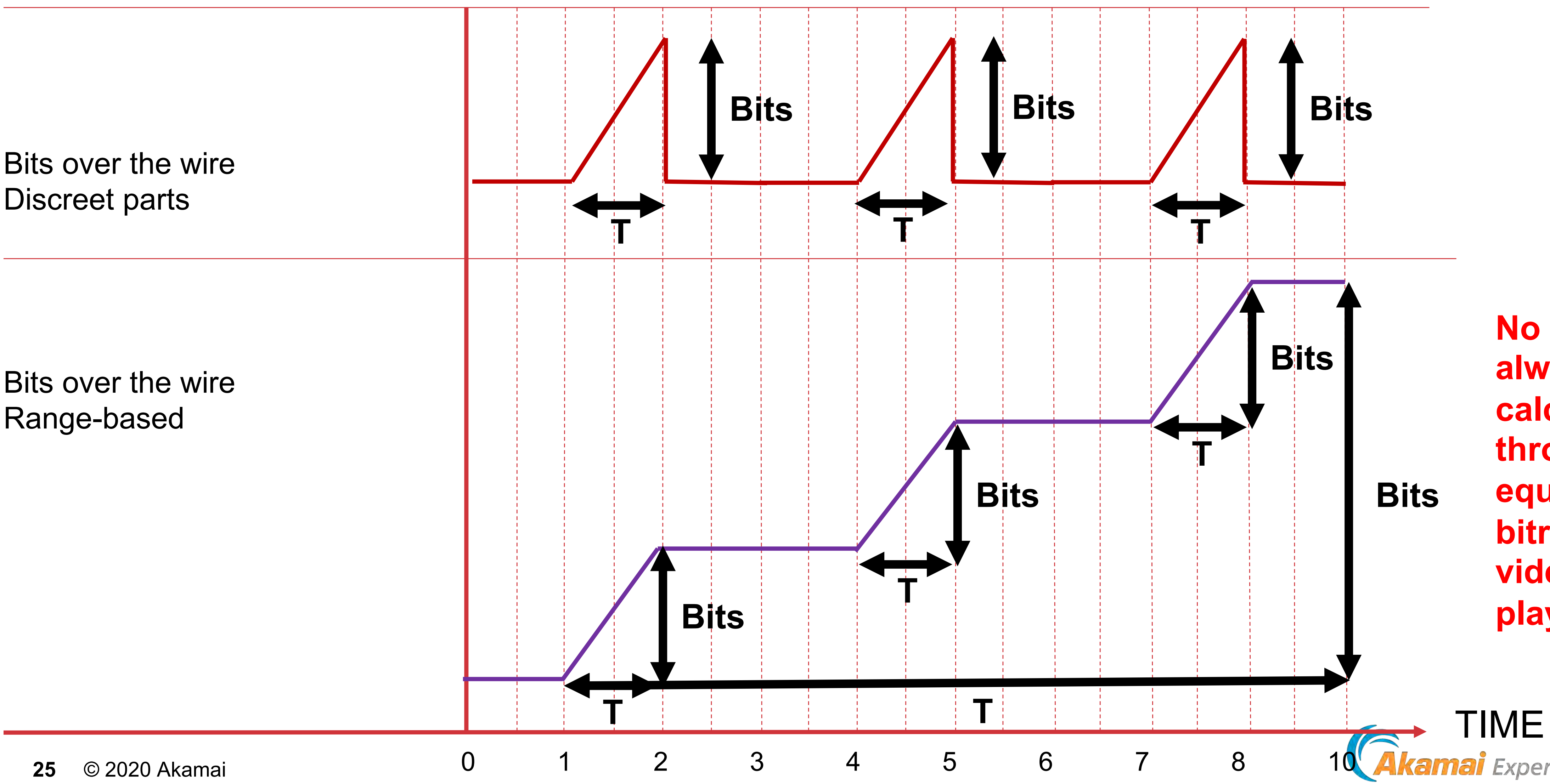
As long as it begins playback at the beginning of a segment, a LL-HLS client can play a byte-range addressed playlist without making any range requests.

Experience the Edge

Request rate improvements

Mode	Requests per segment duration					Gain
	Video playlist	Audio playlist	Video segment/Part	Audio segment/part	Total	% Reduction
4s segment 1s part Discreet	4	4	4	4	16	0
4s segment 1s part Range-based	4	4	1	1	10	37.5%
4s segment 0.5s part Discreet	8	8	8	8	32	0
4s segment 0.5s part Range-based	8	8	1	1	18	43%

A note on bandwidth estimation



No – this will always calculate the throughout as equal to the bitrate of the video you are playing.

Akamai Experience the Edge

Summary: Byte-Range addressing

Byte range addressing with LL-HLS offers

- Increased cache efficiency at origin and CDN distribution tiers.
- Decreased request rate from clients
- Interoperability with standard latency HLS clients, low latency DASH clients and standard latency DASH clients.
- It requires support for RFC8673 at the origin, CDN and client layers to work effectively with mid-segment range requests.

Summary: low latency streaming

- Both LL-DASH and LL-HLS offer solutions to achieve scalable streams with segmented media in the 2s+ range
- Both solutions rely upon chunked encoding of the media content and are able to decouple end-to-end latency from segment duration.
- LL-DASH, and optionally LL-HLS, rely upon the transfer of aggregating objects. These complicate delivery networks as the content-length is not known and hence multi-tiered systems have a hard time reacting to corrupt or slow sessions.
- RFC 8673 is necessary to disambiguate the desired response mode when range-requests are made against an aggregating object.
- Byte-range addressing mode in LL-HLS offers a cache-efficient solution when combined with LL-DASH.

Live Demo and Q&A

LL-DASH and LL-HLS playing from a common origin in London over Akamai CDN to clients in San Francisco with 2.5s of E2E latency.